

FILEID**SATSSS43

J 5

SAT
V04

A musical score page featuring various clefs and note heads. The top section contains eight measures of music, each consisting of a single note. The notes are represented by different characters: 'S' for a whole note, 'A' for an eighth note, 'T' for a sixteenth note, and '3' for a thirty-second note. The bottom section contains two measures of music, also consisting of single notes represented by the same characters.

(1)	54	DECLARATIONS
(1)	77	MACROS
(1)	129	R/W PSECT
(1)	210	SATSSS43
(1)	261	DCLCMH TESTS
(1)	358	DCLEXH TESTS #1
(1)	412	CANEXH TESTS
(1)	458	DCLEXH TESTS #2
(2)	522	SUPER MODE
(2)	587	USER_MODE
(2)	616	COMP_MODE
(2)	648	SETUP SUPER ROUTINE
(2)	728	REG_SAVE
(2)	750	REG_CHECK
(2)	793	REG_CHECKNP
(2)	857	ERLBUF_DUMP
(2)	895	PRINT_FAIL
(2)	952	MODE_ID

0000 1 .TITLE SATSSS43 - SATS SYSTEM SERVICE TESTS (SUCC S.C.)
0000 2 .IDENT 'V04-000'
0000 3 .
0000 4 .
0000 5 *****
0000 6 .
0000 7 .* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 .* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 .* ALL RIGHTS RESERVED.
0000 10 .
0000 11 .* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 .* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 .* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 .* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 .* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 .* TRANSFERRED.
0000 17 .
0000 18 .* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 .* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 .* CORPORATION.
0000 21 .
0000 22 .* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 .* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 .
0000 25 .
0000 26 *****
0000 27 .
0000 28 .
0000 29 .++
0000 30 : FACILITY: SATS SYSTEM SERVICE TESTS
0000 31 : ABSTRACT: The SATSSS43 module tests the execution of the following
0000 32 : VMS system services:
0000 33 :
0000 34 :
0000 35 : \$DCLCMH
0000 36 : \$DCLEXH
0000 37 : \$SCANEXH
0000 38 :
0000 39 : ENVIRONMENT: User, Supervisor and Executive mode image.
0000 40 : Needs CMKRNLL privilege and dynamically acquires other
0000 41 : privileges, as needed.
0000 42 :
0000 43 : AUTHOR: THOMAS L. CAFARELLA, CREATION DATE: MMM, 1978
0000 44 : PAUL D. FAY (DISPSERV & TESTSERV MACROS)
0000 45 :
0000 46 : MODIFIED BY:
0000 47 :
0000 48 :
0000 49 : V03-001 LDJ0001 Larry D. Jones 17-Sep-1980
0000 50 : Modified to conform to new build command procedures.
0000 51 :**
0000 52 :--

```
0000 54 .SBTTL DECLARATIONS
0000 55 :
0000 56 : MACRO LIBRARY CALLS
0000 57 :
0000 58 : $PCBDEF : PCB definitions
0000 59 : $PHDDEF : process header definitions
0000 60 : $PRDEF : processor register definitions
0000 61 : $PRVDEF : privilege definitions
0000 62 : $PSLDEF : PSL definitions
0000 63 : $SFDEF : Stack Frame definitions
0000 64 : $SHR MESSAGES UETP,116,<<TEXT,INFO>> ; UETP$ TEXT definition
0000 65 : $SSDEF : system service definitions
0000 66 : $STSDEF : STS definitions
0000 67 : $UETPDEF : UETP message definitions
0000 68 :
0000 69 : Equated symbols
0000 70 :
0000 71 WARNING = 0 : warning severity value for msgs
0000 72 SUCCESS = 1 : success "
0000 73 ERROR = 2 : error "
0000 74 INFO = 3 : information "
0000 75 SEVERE = 4 : fatal "
0000 76 :
0000 77 .SBTTL MACROS
0000 78 :
0000 79 .MACRO EHDB MODE,NUM
0000 80 .LIST MEB
0000 81 MODE'NUM:
0000 82 .LONG 0
0000 83 .ADDRESS MODE'H'NUM
0000 84 .LONG 2
0000 85 .ADDRESS STATUS
0000 86 .LONG NUM
0000 87 .NLIST MEB
0000 88 .ENDM EHDB
0000 89 :
```

```

00000000 91 .PSECT RODATA,RD,NOWRT,NOEXE,LONG
0000 92 .
0000 93 TEST_MOD_NAME:
0000 94 .ASCIIC /SATSSS43/ ; needed for SATSMS message
0009 95 TEST_MOD_NAME_D:
0009 96 .ASCIID /SATSSS43/ ; module name
0017 97 TEST_MOD_BEGIN:
0019 98 .ASCIIC /begun/
001F 99 TEST_MOD_SUCC:
001F 100 .ASCIIC /successful/
002A 101 TEST_MOD_FAIL:
002A 102 .ASCIIC /failed/
0031 103 DCLCMH:
0031 104 .ASCIIC /DCLCMH/
0038 105 DCLEXH:
0038 106 .ASCIIC /DCLEXH/
003F 107 CANEXH:
003F 108 .ASCIIC /CANEXH/
0046 109 CS1:
0046 110 .ASCID \Test !AC service name !AC step !UL failed.\

0078 111 CS2:
0078 112 .ASCID \Expected !AS = !XL received !AS = !XL\

00A5 113 CS3:
00A5 114 .ASCID \Expected !AS!UB = !XL received !AS!UB = !XL\

00D8 115 CS4:
00D8 116 .ASCID \Unexpected !AS mode exit handler found in !AS.\

010E 117 CS5:
010E 118 .ASCID \Mode was !AS.\

011C 119 UM:
0123 120 .ASCID \user\
012F 121 SM:
012F 122 .ASCID \super\
013C 123 EM:
013C 124 .ASCID \executive\

```

SATSSS43
V04-000

- SATS SYSTEM SERVICE TESTS (SUCC S.C.) 16-SEP-1984 00:54:19 VAX/VMS Macro V04-00
MACROS 5-SEP-1984 04:31:29 [UETPSY.SRC]SATSSS43.MAR;1 Page 4
(1)

65 76 69 014A
014D 125 EXP:
73 75 74 61 74 73 00000155'010E0000' 014D 126 .ASCID \status\

SAT
V04

015B 128 :
 015B 129 : .SBTTL R/W PSECT
 00000000 130 : .PSECT RWDATA,RD,WRT,NOEXE,LONG
 0000 131 :
 0000 132 : PID:
 00000000 0000 133 : LONG 0 ; PID for this process
 0004 134 CURRENT_TC: .LONG 0 ; ptr to current test case
 00000000 0004 135 : .LONG 0
 0008 136 : .ALIGN LONG
 0008 137 REG_SAVE_AREA: .BLKL 15 ; register save area
 00000044 0008 138 : .MOD_MSG_CODE: .LONG UETPS\$_SATSMS ; test module message code for putmsg
 007480D9 0044 140 : .TMN_ADDR: .ADDRESS TEST_MOD_NAME
 00000000' 0048 141 : .TMD_ADDR: .ADDRESS TEST_MOD_BEGIN
 00000019' 004C 144 :
 0050 145 PRVPRT: .BYTE 0 ; protection return byte for SETPRT
 0051 146 :
 00000000 00000000 0051 147 PRIVMASK: .QUAD 0 ; priv. mask
 0059 148 :
 00000000 0059 149 CHM_CONT: .LONG 0 ; change mode continue address
 005D 150 :
 00000065 005D 151 RETADR: .BLKL 2 ; returned address's from SETPRT
 0065 152 :
 00000000 0065 153 STATUS: .LONG 0
 0069 154 :
 00000000 0069 155 MODE: .LONG 0
 006D 156 :
 006D 157 DCL: \$DCLCMH DUMMY,OHC,0 ; DCLCMH parameter list
 007D 158 :
 007D 159 DCL1: \$DCLEXH EXEC3 ; DCLEXH parameter list
 0085 160 :
 0085 161 CAN: \$CANEXH EXEC1 ; CANEXH parameter list
 008D 162 :
 008D 163 REG: .ASCID \register R\
 74 73 69 67 65 72 00000095' 010E0000' 164 :
 52 20 72 65 009B :
 009F 165 REGNUM: .LONG 0 ; register number
 00000000 009F 166 :
 00A3 167 MSGL: .LONG 80 ; buffer desc.
 00000050 00A3 168 :
 000000AB' 00A7 169 : ADDRESS BUF
 00AB 170 BUF: .BLKB 80
 000000FB 00AB 171 :
 00FB 172 MESSAGE: .LONG 0 ; message desc.
 00000000 00FB 173 :
 000000AB' 00FF 174 : ADDRESS BUF
 0103 175 SERV_NAME: .LONG 0 ; service name pointer
 00000000 0103 176 :
 0107 177 PRVHND1: .LONG 0 ; previous handler address 1
 00000000 0107 178 :
 010B 179 PRVHND2: .LONG 0 ; previous handler address 2
 00000000 010B 180 :
 010F 181 PRVHND3: .LONG 0 ; previous handler address 3
 00000000 010F 182 :
 0113 183 OHC: .LONG 0

00000000	0113	184	.LONG	0	; old handler check location
	0117	185	ARGLST:		
00000001	0117	186	.LONG	1	; super mode setup arg list
000003E6	011B	187	:ADDRESS	SUPER_MODE	
	011F	188	MSGVEC:		; PUTMSG message vector
00000003	011F	189	.LONG	3	
00741133	0123	190	.LONG	UETPS_TEXT	
00000001	0127	191	.LONG	1	
000000FB	012B	192	:ADDRESS	MESSAGE1	
	012F	193	MSGVEC1:		
00000004	012F	194	.LONG	4	; PUTMSG message vector for exit
00000000	0133	195	.LONG	0	
00000002	0137	196	.LONG	2	
00000143	013B	197	.BLKL	2	

0143 199 ; exit handler desc. blocks
0143 200 EHDB USER,1 ; user #1 will be deleted
0143 USER1:
00000000 0143 .LONG 0
00000735 0147 .ADDRESS USERH1
00000002 014B .LONG 2
00000065 014F .ADDRESS STATUS
00000001 0153 .LONG 1
0157 201 EHDB USER,2 ; user #2 will be used
0157 USER2:
00000000 0157 .LONG 0
00000353 015B .ADDRESS USERH2
00000002 015F .LONG 2
00000065 0163 .ADDRESS STATUS
00000002 0167 .LONG 2
016B 202 EHDB USER,3 ; user #3 will be deleted
016B USER3:
00000000 016B .LONG 0
00000735 016F .ADDRESS USERH3
00000002 0173 .LONG 2
00000065 0177 .ADDRESS STATUS
00000003 017B .LONG 3
017F 203 EHDB USER,4 ; user #4 will be used
017F USER4:
00000000 017F .LONG 0
0000034C 0183 .ADDRESS USERH4
00000002 0187 .LONG 2
00000065 018B .ADDRESS STATUS
00000004 018F .LONG 4
0193 204 EHDB SUPER,1 ; super #1 will be deleted
0193 SUPER1:
00000000 0193 .LONG 0
00000740 0197 .ADDRESS SUPERH1
00000002 019B .LONG 2
00000065 019F .ADDRESS STATUS
00000001 01A3 .LONG 1
01A7 205 EHDB SUPER,3 ; super #3 will be deleted
01A7 SUPER3:
00000000 01A7 .LONG 0
00000740 01AB .ADDRESS SUPERH3
00000002 01AF .LONG 2
00000065 01B3 .ADDRESS STATUS
00000003 C1B7 .LONG 3
01BB 206 EHDB EXEC,1 ; exec #1 will be deleted
01BB EXEC1:
00000000 01BB .LONG 0
0000074B 01BF .ADDRESS EXECH1
00000002 01C3 .LONG 2
00000065 01C7 .ADDRESS STATUS
00000001 01CB .LONG 1
01CF 207 EHDB EXEC,3 ; exec #3 will be deleted
01CF EXEC3:
00000000 01CF .LONG 0
0000074B 01D3 .ADDRESS EXECH3
00000002 01D7 .LONG 2
00000065 01DB .ADDRESS STATUS
00000003 01DF .LONG 3

00000000 209 .PSECT SATSSS43, RD,WRT,EXE, LONG
0000 210 :SBTTL SATSSS43
0000 211 :++
0000 212 : FUNCTIONAL DESCRIPTION:
0000 213 :
0000 214 : After performing some initial housekeeping, such as
0000 215 : printing the module begin message and acquiring needed privileges,
0000 216 : the system services are tested in each of their normal conditions.
0000 217 : Detected failures are identified and an error message is printed
0000 218 : on the terminal. Upon completion of the test a success or fail
0000 219 : message is printed on the terminal.
0000 220 :
0000 221 : CALLING SEQUENCE:
0000 222 :
0000 223 : \$ RUN SATSSS43 ... (DCL COMMAND)
0000 224 :
0000 225 : INPUT PARAMETERS:
0000 226 :
0000 227 : none
0000 228 :
0000 229 : IMPLICIT INPUTS:
0000 230 :
0000 231 : none
0000 232 :
0000 233 : OUTPUT PARAMETERS:
0000 234 :
0000 235 : none
0000 236 :
0000 237 : IMPLICIT OUTPUTS:
0000 238 :
0000 239 : Messages to SYSS\$OUTPUT are the only output from SATSSS43.
0000 240 : They are of the form:
0000 241 :
0000 242 : XUETP-S-SATSMS, TEST MODULE SATSSS43 BEGUN ... (BEGIN MSG)
0000 243 : XUETP-S-SATSMS, TEST MODULE SATSSS43 SUCCESSFUL ... (END MSG)
0000 244 : XUETP-E-SATSMS, TEST MODULE SATSSS43 FAILED ... (END MSG)
0000 245 : XUETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
0000 246 :
0000 247 : COMPLETION CODES:
0000 248 :
0000 249 : The SATSSS43 routine terminates with a \$EXIT to the
0000 250 : operating system with a status code defined by UETPS\$_SATSMS.
0000 251 :
0000 252 : SIDE EFFECTS:
0000 253 :
0000 254 : none
0000 255 :
0000 256 :--
0000 257 :
0000 258 :
0000 259 :
0000 260 TEST_START SATSSS43 ; let the test begin

```

0004'CF 0000 0000 .ENTRY SATSSS43,0
0000 00 DD 0002 CLRL W^CURRENT_TC
0000 00 DF 0006 PUSHL #0
0000 0000'GF 02 FB 000C PUSHAL W^TPID
0000 0000'GF 00 FB 0013 CALLS #2,G^SYSSWAKE
0000 0009'GF 01 7F 001A CALLS #0,G^SYSSHIBER
0000 0000'GF 01 FB 001E PUSHAQ W^TEST_MOD_NAME_D
0044'CF 001F'CF 0798 30 0025 CALLS #1,G^SYSSSETPRN
0044'CF 00 01 FO 002F BSBW W^MOD_MSG_PRINT
0044'CF 00 00 DD 0036 MOVAL W^TEST_MOD_SUCC,W^TMD_ADDR
0549'CF 01 FB 0038 INSV #SUCCESS,#0,#3,W^MOD_MSG_CODE
003D STP0: PUSHL #0
003D CALLS #1,W^REG_SAVE

261 003D .SBTTL DCLCMH TESTS
262 003D :+
263 003D :-
264 003D : $DCLCMH tests
265 003D : test super mode handler declaration
266 003D :-
267 003D :-
268 0103'CF 0031'CF DE 003D MOVAL W^DCLCMH,W^SERV_NAME ; set service name
269 0044 0044 $CMKRNL_S W^SETUP_SUPER,W^ARGLST ; test super mode declaration
270 0663'CF 00 FB 0053 CALLS #0,W^ERLBUF_DUMP ; report any errors
271 01 BE 0058 CHMS #1 ; declare dummy handler
272 0113'CF 000003E6'8F D1 005A CMPL #SUPER_MODE,W^OHC ; make sure it happened
273 11 13 0063 BEQL 10$ ; br if yes
274 0113'CF DD 0065 PUSHL W^OHC ; else setup to report the error
275 03E6'CF DF 0069 PUSHAL W^SUPER_MODE ; save the expected results
276 014D'CF DF 006D PUSHAL W^EXP ; push the message address
277 0691'CF 03 FB 0071 CALLS #3,W^PRINT_FAIL ; report the failure
278 0076 10$: 02 BE 0076 CHMS #2 ; remove the dummy handler
279 0078 :+
280 0078 :-
281 0078 : test user mode handler declaration
282 0078 :-
283 0078 :-
284 0078 :-
285 0078 NEXT_TEST
0078 STP1:
0004'CF 01 DO 0078 MOVL #1,W^CURRENT_TC
0000 00 DD 007D PUSHL #0
0549'CF 01 FB 007F CALLS #1,W^REG_SAVE
0069'CF 0123'CF DE 0084 286 MOVAL W^UM,W^MODE ; set the mode
0071'CF 049A'CF DE 008B 287 MOVAL W^DUMMY,W^DCL+DCLCMHS_ADDRES ; reset the handler address
0075'CF 010B'CF DE 0092 288 MOVAL W^PRVHND2,W^DCL+DCLCMHS_PRVHND ; set new handler save address
0099 0099 00A2 289 SDCLCMH G W^DCL ; check G form
00A2 00A2 290 FAIL_CHECK SSS_NORMAL ; check for success
0553'CF 01 DD 00A2 00A4 PUSHL #SSS_NORMAL
0553'CF 01 FB 00A4 291 CALLS #1,W^REG_CHECK ; set real handler
00BA 00BA 292 SDCLCMH S W^USER_MODE,W^OHC ; check for success
00BC 00BC 293 FAIL_CHECK SSS_NORMAL
0113'CF 0000049A'8F D1 00C1 293 PUSHL #SSS_NORMAL
CMPL #DUMMY,W^OHC ; is handler address correct?

```

0113'CF	11	13	00CA	294	BEQL	10\$; br if yes
049A'CF	DD	00CC	295	PUSHL	W^OHC	; push received address	
014D'CF	DF	00D0	296	PUSHAL	W^DUMMY	; push expected address	
0691'CF	03	FB	00D4	297	PUSHAL	W^EXP	; push string variable
			00D8	298	CALLS	#3,W^PRINT_FAIL	; print the error
			00DD	299	10\$:		
			00DD	300	:		
			00DD	301	:		
			00DD	302	: test for compatibility mode handler declaration		
			00DD	303	:		
			00DD	304	-		
			00DD	305	NEXT_TEST		
			00DD				
			00DD		STP2:		
0004'CF	02	DO	00DD		MOVL	#2,W^CURRENT_TC	
00	00	DD	00E2		PUSHL	#0	
0549'CF	01	FB	00E4		CALLS	#1,W^REG_SAVE	
0075'CF	010F'CF	DE	00E9	306	MOVAL	W^PRVHND3,W^DCL+DCLCMHS_PRVHND	; set new handler save location
0079'CF	D6	00F0	307	INCL	W^DCL+DCLCMHS_TYPE	; set to compatibility mode type	
	00F4	308	\$DCLCMH	G W^DCL	; check G form		
	00FD	309	FAIL_CHECK	SSS_NORMAL	; check for success		
0553'CF	01	DD	00FD		PUSHL	SSS_NORMAL	
	01	FB	00FF		CALLS	#1,W^REG_CHECK	
	0104	310	\$DCLCMH	S W^COMP_MODE,W^OHC,#1			
	0115	311	FAIL_CHECK	SSS_NORMAL			
	01	DD	0115		PUSHL	SSS_NORMAL	
0553'CF	01	FB	0117		CALLS	#1,W^REG_CHECK	
0000049A'8F	D1	011C	312	CMPL	#DUMMY,W^OHC	; is handler address correct?	
11	13	0125	313	BEQL	10\$; br if yes	
0113'CF	DD	0127	314	PUSHL	W^OHC	; push received address	
049A'CF	DF	012B	315	PUSHAL	W^DUMMY	; push expected address	
014D'CF	DF	012F	316	PUSHAL	W^EXP	; push string variable	
0691'CF	03	FB	0133	317	CALLS	#3,W^PRINT_FAIL	; print the error
	0138	318	10\$:				
	0138	319	:				
	0138	320	-				
	0138	321	: check the compatibility mode handler				
	0138	322	:				
	0138	323	-				
	0138	324	NEXT_TEST				
	0138						
	0138						
	0138						
	0138						
0004'CF	03	DO	0138		STP3:		
00	00	DD	013D		MOVL	#3,W^CURRENT_TC	
0549'CF	01	FB	013F		PUSHL	#0	
83C00000 8F	DD	0144	325	CALLS	#1,W^REG_SAVE		
	014A	326	PUSHL	#<<PSL\$M CM>!<PSE\$C USER@PSL\$V_PRVMOD>-			
	4E'AF	DF	014A	327		!<PSL\$C_USER@PSL\$V_CURMOD>>	; set compatibility mode
	02	014D	328	PUSHAL	B^10\$; set new address	
				REI		; enter compatibility mode	

```

      014E 330 .ALIGN WORD ; adjust addressing for PDP-11's
      014E 331 10$: .WORD ^0012767 ; MOV #-1,TEST ;prove we were here
  15F7 014E 332 .WORD ^0177777
  FFFF 0150 333 .WORD ^0000002 ; HALT ;cause an exception
  0002 0152 334 .WORD ^0000000
  0000 0154 335 .WORD ^0000000
      0156 336 TEST: .WORD ^0000000 ; compatibility mode flag location
      0000 0156 337 .WORD ^0000000 ; return to the good life
      0158 338 RETURN: .WORD ^0000000
      0158 339 :+
      0158 340 :
      0158 341 : test the user mode handler
      0158 342 :
      0158 343 :
      0158 344 NEXT_TEST
      0158
  0004'CF 04 DD 0158 STP4:
      00 DD 015D MOVL #4,W^CURRENT_TC
  0549'CF 01 FB 015F PUSHL #0
      05 BF 0164 CALLS #1,W^REG_SAVE ; use a param of 5
      0166 345 CHMU #5
      0166 346 :+
      0166 347 :
      0166 348 : reset handlers to the original address
      0166 349 :
      0166 350 :
      0166 351 NEXT_TEST
      0166
  0004'CF 05 DD 0166 STP5:
      00 DD 016B MOVL #5,W^CURRENT_TC
  0549'CF 01 FB 016D PUSHL #0
      0103'CF 0031'CF DE 0172 352 CALLS #1,W^REG_SAVE ; set service name
      0172 353 MOVAL W^DCLCMH,W^SERV_NAME ; reset CHMU handler
      0179 354 $DCLCMH S 0,W^PRVHND2 ; check for success
      0188 355 FAIL_CHECK $SS_NORMAL
      018A 356 PUSHL #SSS_NORMAL
      018F 355 CALLS #1,W^REG_CHECK ; reset CM handler
      019E 356 $DCLCMH S 0,W^PRVHND3,#1 ; check for success
      019E 356 FAIL_CHECK $SS_NORMAL
      01A0 356 PUSHL #SSS_NORMAL
      01A0 356 CALLS #1,W^REG_CHECK

```

01A5 358 .SBTTL DCLEXH TESTS #1
 01A5 359 :+
 01A5 360 : SDCLEXH tests
 01A5 361 : These tests are divided into two parts. This part is the declaration
 01A5 362 : tests. The second part is the servicing part.
 01A5 363 : test for exec mode exit handler declaration
 01A5 364 :
 01A5 365 :
 01A5 366 :
 01A5 367 :
 01A5 368 :-
 01A5 369 : NEXT_TEST
 01A5 :
 0004'CF 06 DD 01A5 STP6:
 00 00 DD 01AA MOVL #6,W^CURRENT_TC
 0549'CF 01 FB 01AC PUSHL #0
 0069'CF 013C'CF DE 01B1 CALLS #1,W^REG_SAVE
 0103'CF 0038'CF DE 01B8 MOVAL W^EM,W^MODE : set the mode
 01B8 01BF SCLEXH,S B^10\$: set service name
 2C 11 01CB \$CMEXEC_S B^10\$: get to exec mode
 01CD 01CD BRB 20\$: skip over exec routine
 0000 01CD 374 10\$: .WORD 0
 0549'CF 00 DD 01CF 375 PUSHL #0 : push a dummy parameter
 01 FB 01D1 376 CALLS #1,W^REG_SAVE : save a reg snapshot
 01D6 01E1 SCLEXH,S W^EXEC : declare #1 exec exit handler
 01E1 379 FAIL_CHECKNP SSS NORMAL : check for success
 05EA'CF 01 DD 01E1 PUSHL #SSS NORMAL
 01 FB 01E3 CALLS #1,W^REG_CHECKNP : declare #3 exec exit handler
 01E8 01F1 380 SCLEXH,G W^DCL1 : declare #3 exec exit handler
 01F1 381 FAIL_CHECKNP SSS NORMAL : check for success
 05EA'CF 01 DD 01F1 PUSHL #SSS NORMAL
 01 FB 01F3 CALLS #1,W^REG_CHECKNP : go back to user mode
 04 01F8 382 RET :
 0663'CF 00 FB 01F9 383 20\$: CALLS #0,W^ERLBUF_DUMP : dump any errors that occurred
 01FE 384 :+
 01FE 385 : test super mode exit handler declaration
 01FE 386 :
 01FE 387 :
 01FE 388 :
 01FE 389 : -
 01FE 390 : NEXT_TEST
 01FE :
 0004'CF 07 DD 01FE STP7:
 00 00 DD 0203 MOVL #7,W^CURRENT_TC
 0549'CF 01 FB 0205 PUSHL #0
 0069'CF 012F'CF DE 020A CALLS #1,W^REG_SAVE
 04 BE 0211 391 MOVAL W^SM,W^MODE : set the mode
 392 CHMS #4 : declare 2 super mode exit handlers

0213 394 ;+
0213 395 ;:
0213 396 ; test user mode exit handler declaration
0213 397 ;-
0213 398 ;-
0213 399 NEXT_TES1
0213 STP8:
0004'CF 08 DD 0213 MOVL #8,W^CURRENT_TC
00 DD 0218 PUSHL #0
0549'CF 01 FB 021A CALLS #1,W^REG_SAVE
0069'CF 0123'CF DE 021F 400 MOVAL W^UM,W^MODE : set the mode
0226 401 \$DCLEXH S W^USER1 : declare #1 user mode exit handler
0231 402 FAIL_CHECK SSS_NORMAL : check for success
0553'CF 01 DD 0231 PUSHL #SSS_NORMAL
01 FB 0233 CALLS #1,W^REG_CHECK
0081'CF 0157'CF DE 0238 403 MOVAL W^USER2,W^DCL1+\$DCLEXH\$_DESBLOCK ; set exit handler address
023F 404 \$DCLEXH G W^DCL1 : declare #2 user mode exit handler
0248 405 FAIL_CHECK SSS_NORMAL : check for success
0553'CF 01 DD 0248 PUSHL #SSS_NORMAL
01 FB 024A CALLS #1,W^REG_CHECK
024F 406 \$DCLEXH S W^USER3 : declare #3 user mode exit handler
025A 407 FAIL_CHECK SSS_NORMAL : check for success
0553'CF 01 DD 025A PUSHL #SSS_NORMAL
01 FB 025C CALLS #1,W^REG_CHECK
0081'CF 017F'CF DE 0261 408 MOVAL W^USER4,W^DCL1+\$DCLEXH\$_DESBLOCK ; set exit handler address
0268 409 \$DCLEXH G W^DCL1 : declare #4 user mode exit handler
0271 410 FAIL_CHECK SSS_NORMAL : check for success
0553'CF 01 DD 0271 PUSHL #SSS_NORMAL
01 FB 0273 CALLS #1,W^REG_CHECK

```

0278 412 .SBTTL CANEXH TESTS
0278 413 :+
0278 414 :-
0278 415 : SCANEXH tests
0278 416 : test for exec mode exit handler deletion
0278 417 :-
0278 418 :-
0278 419 NEXT_TEST
0278
0278 STP9:
0004'CF 09 DO 0278 MOVL #9,W^CURRENT_TC
0549'CF 00 DD 0270 PUSHL #0
0069'CF 013C'CF DE 0284 420 MOVAL W^EM,W^MODE
0103'CF 003F'CF DE 028B 421 MOVAL W^CANEXH,W^SERV_NAME ; set the mode
013C'CF 33 11 029E 422 SCMEXEC S B^10$ ; set service name
0103'CF 02A0 423 BRB -20$ ; get to exec mode
0549'CF 0000 02A0 424 10$: ; skip over the routine
01 00 DD 02A2 425 .WORD 0 ; entry mask
0549'CF 01 FB 02A4 426 PUSHL #0 ; push a dummy parameter
02A9 427 CALLS #1,W^REG_SAVE ; save a reg snapshot
02A9 428 SCANEXH S W^EXEC ; cancel exec exit handler #1
02B4 429 FAIL_CHECKNP SSS NORMAL ; check for success
05EA'CF 01 DD 02B4 PUSHL #SSS NORMAL
01 CF'CF 01 FB 02B6 430 CALLS #1,W^REG_CHECKNP
0089'CF 02BB 431 MOVAL W^EXEC3,W^CAN+CANEXHS_DESBLK ; set handler adr
01CF'CF 02C2 432 SCANEXH G W^CAN ; cancel exec exit handler #3
02CB 432 FAIL_CHECKNP SSS NORMAL ; check for success
05EA'CF 01 DD 02CB PUSHL #SSS NORMAL
01 FB 02CD 433 CALLS #1,W^REG_CHECKNP ; return
04 02D2 434 20$: RET
0663'CF 00 FB 02D3 435 CALLS #0,W^ERLBUF_DUMP ; dump any errors that occurred
02D8 436 :+
02D8 437 :-
02D8 438 : test super mode exit handler cancellation
02D8 439 :-
02D8 440 :-
02D8 441 NEXT_TEST
02D8
02D8 STP10:
0004'CF 0A DO 02D8 MOVL #10,W^CURRENT_TC
0549'CF 00 DD 02DD PUSHL #0
0069'CF 012F'CF 01 FB 02DF 442 MOVAL W^SM,W^MODE ; set the mode
012F'CF 05 BE 02E4 443 CHMS #5 ; cancel super exit handlers #1 and #3

```

02ED 445 :+
02ED 446 :
02ED 447 : test user mode exit handler cancellation
02ED 448 :
02ED 449 :-
02ED 450 NEXT_TEST
02ED STP11:
0004'CF 0B DO 02ED MOVL #11,W^CURRENT_TC
0000 DD 02F2 PUSHL #0
0549'CF 01 FB 02F4 CALLS #1,W^REG_SAVE
0069'CF 0123'CF DE 02F9 451 MOVAL W^UM,W^MODE ; set the mode
0300 452 SCANEXH S W^USER1 ; cancel user exit handler #1
030B 453 FAIL_CHECK SSS_NORMAL ; check for success
0553'CF 01 DD 030B PUSHL #SSS_NORMAL
0089'CF 016B'CF FB 030D CALLS #1,W^REG_CHECK
DE 0312 454 MOVAL W^USER3,W^CAN+CANEXHS_DESBLK ; set handler adr
0319 455 SCANEXH G W^CAN ; cancel user exit handler #3
0322 456 FAIL_CHECK SSS_NORMAL ; check for success
0553'CF 01 DD 0322 PUSHL #SSS_NORMAL
FB 0324 CALLS #1,W^REG_CHECK

```

0329 458 .SBTTL CCLEXH TESTS #2
0329 459 ;+
0329 460 ; $DCLEXH tests
0329 461 ; This is the second of two parts of the DCLEXH tests.
0329 462 ; This part tests the servicing of the exit handlers.
0329 463 ; At this time there should be 2 user mode exit handlers declared.
0329 464 ; test user mode exit handler #4
0329 465 ;-
0329 466 ; NEXT_TEST
0329 467 ;+
0329 468 ;-
0329 469 ;-
0329 470 ;+
0329 471 ; STP12:
0329 472 ; MOVL #12,W^CURRENT_TC
0329 473 ; PUSHL #0
0329 474 ; CALLS #1,W^REG_SAVE
0329 475 ; MOVAL W^DCLEXH,W^SERV_NAME : set service name
0329 476 ; MOVL S^$SS NORMAL,W^STATUS : set the expected status return
0329 477 ; $EXIT_S W^MOD_MSG_CODE : kick off ALL exit handlers
0329 478 ;+
0329 479 ; test user exit handler #2
0329 480 ;-
0329 481 ;+
0329 482 ;-
0329 483 ; USERH2:
0329 484 ; WORD 0
0329 485 ; NEXT_TEST
0329 486 ;+
0329 487 ; STP13:
0329 488 ; MOVL #13,W^CURRENT_TC
0329 489 ; PUSHL #0
0329 490 ; CALLS #1,W^REG_SAVE
0329 491 ; MOVZBL S^#2,R2 ; set expected handler code
0329 492 ; CMPL @B^4(AP),W^STATUS ; is the status adr field OK?
0329 493 ; BEQL 10$ ; br if yes
0329 494 ; PUSHL 4(AP) ; push received code
0329 495 ; PUSHAL W^STATUS ; push expected code
0329 496 ; PUSHAL W^EXP ; push string variable
0329 497 ; CALLS #3,W^PRINT_FAIL ; print the error
0329 498 ; CALLS #0,W^MODE_ID ; identify the handler mode
0329 499 ; 10$: CMPL R2,8(AP) ; is the argument field OK?
0329 500 ; BEQL 20$ ; br if yes
0329 501 ; PUSHL 8(AP) ; push received code
0329 502 ; PUSHL R2 ; push expected code
0329 503 ; PUSHAL W^EXP ; push string variable
0329 504 ; CALLS #3,W^PRINT_FAIL ; print the error
0329 505 ; CALLS #0,W^MODE_ID ; identify the exit handler mode
0329 506 ; 20$: CMPB S^#2,8(AP) ; is this the last handler?

```

8 7

01	13	039E	505		BEQL	30\$	
	04	03A0	506		RET		: br if yes
		03A1	507	30\$: :			; do the next handler
00	DD	03A1	508		PUSHL	#0	
0549'CF	01	FB	03A3	509	CALLS	#1,W^REG_SAVE	: push dummy parameter
0069'CF	012F'CF	DE	03A8	510	MOVAL	W^SM,W^MODE	: save the registers
	03	BE	03AF	511	CHMS	#3	: set the mode
0133'CF	0044'CF	DO	03B1	512	MOVL	W^MOD_MSG_CODE,W^MSGVEC1+4	: reset the CHMS handler
013B'CF	0048'CF	DO	03B8	513	MOVL	W^TMN_ADDR,W^MSGVEC1+12	: set message code
013F'CF	004C'CF	DO	03BF	514	MOVL	W^TMD_ADDR,W^MSGVEC1+16	: set up parameters
			03C6	515	SPUTMSG_S	W^MSGVEC1	: print the message
1C 01	F0	03D7	516		INSV	#1,#STSSV INHIB_MSG,-	
00000044'EF	01		03DA	517		#1,MOD_MSG_CODE-	: set inhibit printing on the exit status
50 0044'CF		DO	03E0	518	MOVL	W^MOD_MSG_CODE,RO	: save the new code in R0
	04	03E5	519		RET		: leave for good!

03E6 522 .SBTTL SUPER_MODE
 03E6 523 :++
 03E6 524 : FUNCTIONAL DESCRIPTION:
 03E6 525 : Routine to handle the CHMS instructions.
 03E6 526 :
 03E6 527 : CALLING SEQUENCE:
 03E6 528 : CHMS #N
 03E6 529 :
 03E6 530 : INPUT PARAMETERS:
 03E6 531 : SP=> CHMS parameter
 03E6 532 : PC
 03E6 533 : PSL
 03E6 534 :
 03E6 535 : The CHMS parameter can be one of the following:
 03E6 536 :
 03E6 537 : 1 = execute a \$DCLCHM_G to declare a dummy handler
 03E6 538 : 2 = execute a \$DCLCMH_G to clear the dummy CHMS handler
 03E6 539 : 3 = execute a \$DCLCMH_S to reset the CHMS handler
 03E6 540 : 4 = declare 2 exit handlers in super mode
 03E6 541 : 5 = delete 2 exit handlers in super mode
 03E6 542 :
 03E6 543 : OUTPUT PARAMETERS:
 03E6 544 : NONE
 03E6 545 :--
 03E6 546 :
 03E6 547 : SUPER_MODE:
 05 01 50 8E D0 03E6 548 MOVL (SP)+,R0 ; get CHM parameter off the stack
 05 01 50 8F 03E9 549 CASEB R0,#1,#5 ; do the right thing
 03ED 550 10\$:
 000A' 03ED 551 .WORD 20\$-10\$
 001D' 03EF 552 .WORD 30\$-10\$
 0037' 03F1 553 .WORD 40\$-10\$
 0058' 03F3 554 .WORD 50\$-10\$
 0083' 03F5 555 .WORD 60\$-10\$
 03F7 556 20\$:
 03F7 557 \$DCLCMH G W^DCL ; declare a dummy CHMS handler
 0400 558 FAIL_CHECK SSS_NORMAL ; check for success
 0553'CF 01 DD 0400 559 PUSHL #SSS_NORMAL
 008F 01 FB 0402 560 CALLS #1,W^REG_CHECK
 008F 31 0407 561 BRW 70\$; carry on
 0071'CF FFD8 CF DE 040A 562 MOVAL W^SUPER_MODE,W^DCL+DCLCMHS_ADDRES ; set up to delete dummy handler
 0071'CF FFD8 CF DE 0411 563 \$DCLCMH G W^DCL ; clear the dummy handler
 0071'CF FFD8 CF DE 041A 564 FAIL_CHECK SSS_NORMAL ; check for success
 0553'CF 01 DD 041A 565 PUSHL #SSS_NORMAL
 0553'CF 01 FB 041C 566 CALLS #1,W^REG_CHECK
 0075 31 0421 567 BRW 70\$; carry on
 0103'CF 0031'CF DE 0424 568 MOVAL W^DCLCMH,W^SERV_NAME ; set service name pointer
 0103'CF 0031'CF DE 042B 569 \$DCLCMH S @PRVHND1,#0 ; reset the CHMS handler for DCL
 0103'CF 0031'CF DE 043C 570 FAIL_CHECK SSS_NORMAL ; check for success
 0553'CF 01 DD 043C 571 PUSHL #SSS_NORMAL
 0553'CF 01 FB 043E 572 CALLS #1,W^REG_CHECK
 54 11 0443 573 BRB 70\$; carry on
 0445 574 \$DCLEXH S W^SUPER1 ; declare #1 super mode exit handler
 0445 575 FAIL_CHECK SSS_NORMAL ; check for success

```

0553'CF 01 DD 0450      PUSHL #SS$ NORMAL
0553'CF 01 FB 0452      CALLS #1,W^REG_CHECK
0081'CF 01 DE 0457      MOVAL W^SUPER3,W^DCL1+DCLEXHS_DESBLK ; set handler adr for #3
01        045E 573       $DCLEXH G W^DCL1 ; declare #3 super mode exit handler
01        0467 574       FAIL_CHECK SSS_NORMAL ; check for success
0553'CF 01 DD 0467      PUSHL #SS$ NORMAL
0553'CF 01 FB 0469      CALLS #1,W^REG_CHECK
01        29 11 046E 576   BRB 70$ ; carry on
01        0470 577 60$:    SCANEXH S W^SUPER1 ; delete #1 super mode exit handler
01        0470 578         FAIL_CHECK SSS_NORMAL ; check for success
0553'CF 01 DD 0478      PUSHL #SS$ NORMAL
0553'CF 01 FB 047D      CALLS #1,W^REG_CHECK
0089'CF 01 DE 0482      MOVAL W^SUPER3,W^CAN+CANEXHS_DESBLK ; set handler adr for #3
01        0489 580       $CANEXH G W^CAN ; delete #3 super mode exit handler
01        0492 581       FAIL_CHECK SSS_NORMAL ; check for success
0553'CF 01 DD 0492      PUSHL #SS$ NORMAL
0553'CF 01 FB 0494      CALLS #1,W^REG_CHECK
01        0499 583 70$:    0499 584 REI ; go back to user mode
01        049A 585 DUMMY: 049A 586 BRW SUPER_MODE
01        049D 587 .SBTTL USER_MODE ; dummy handler address
01        049D 588 :+++
01        049D 589 : FUNCTIONAL DESCRIPTION:
01        049D 590 : Routine to handle the CHMU instruction
01        049D 591 :
01        049D 592 : CALLING SEQUENCE:
01        049D 593 : CHMU #5
01        049D 594 :
01        049D 595 : INPUT PARAMETERS:
01        049D 596 : SP=> #5
01        049D 597 : PC
01        049D 598 : PSL
01        049D 599 :
01        049D 600 : OUTPUT PARAMETERS:
01        049D 601 : NONE
01        049D 602 :
01        049D 603 :--
01        049D 604 :
01        049D 605 : USER_MODE:
05  8E  D0 049D 606 MOVL (SP)+,R0 ; get CHM parameter off the stack
05  50  D1 04A0 607 CMPL R0,S^#5 ; is it correct?
0D  13  04A3 608 BEQL 10$ ; br if yes
05  DD  04A5 609 PUSHL R0 ; save received
05  DD  04A7 610 PUSHL S^#5 ; save expected
C14D'CF DF 04A9 611 PUSHAL W^EXP ; save the string variable
0691'CF 03  FB 04AD 612 CALLS #3,W^PRINT_FAIL ; print the error message
01        04B2 613 10$:    REI ; return
02  04B2 614

```

04B3 616 .SBTTL COMP_MODE
04B3 617 :++
04B3 618 : FUNCTIONAL DESCRIPTION:
04B3 619 : Compatibility mode exception handler
04B3 620 :
04B3 621 : CALLING SEQUENCE:
04B3 622 : execute a compatibility mode exception
04B3 623 :
04B3 624 : INPUT PARAMETERS:
04B3 625 : NONE
04B3 626 :
04B3 627 : OUTPUT PARAMETERS:
04B3 628 : NONE
04B3 629 :
04B3 630 :--
04B3 631 COMP_MODE:
FC A0 95 04B3 632 TSTB -4(R0) : see if we got the correct exception
OE 13 04B6 633 BEQL 10\$: br if correct
FC A0 DD 04B8 634 PUSHL -4(R0) : push received code
00 DD 04BB 635 PUSHL #0 : push expected code
014D'CF DF 04BD 636 PUSHAL W^EXP : push string variable
0691'CF 03 FB 04C1 637 CALLS #3,W^PRINT_FAIL : print the error
04C6 638 10\$:
FFFF BF FC8C CF B1 04C6 639 CMPW W^TEST,-1 : were we really in compatibility mode?
14 13 04CD 640 BEQL 20\$: br if yes
7E FC83 CF 3C 04CF 641 MOVZWL W^TEST,-(SP) : push received code
0000FFFF 8F DD 04D4 642 PUSHL #^X0000FFFF : push expected code
014D'CF DF 04DA 643 PUSHAL W^EXP : push string variable
0691'CF 03 FB 04DE 644 CALLS #3,W^PRINT_FAIL : print the error
04E3 645 20\$:
FC72 31 04E3 646 BRW RETURN : carry on

04E6 648 .SBTTL SETUP_SUPER ROUTINE
04E6 649 :++
04E6 650 : FUNCTIONAL DESCRIPTION:
04E6 651 : Routine to declare an initial CHMS handler from user mode.
04E6 652 :
04E6 653 : CALLING SEQUENCE:
04E6 654 : \$CMKRNL_S W^SETUP_SUPER,ARGLST
04E6 655 :
04E6 656 : ARGLST = address of a pointer to a one parameter argument list conta
04E6 657 : the address of the entry mask of the CHMS handler
04E6 658 :
04E6 659 : INPUT PARAMETERS:
04E6 660 : ARGLST
04E6 661 :
04E6 662 : IMPLICIT INPUTS
04E6 663 : NONE
04E6 664 :
04E6 665 : OUTPUT PARAMETERS:
04E6 666 : Declares a change mode handler for super mode which must be
04E6 667 : reset to DCL in the users handler routine when the handler is
04E6 668 : no longer needed.
04E6 669 :
04E6 670 : IMPLICIT OUTPUTS:
04E6 671 : NONE
04E6 672 :
04E6 673 : COMPLETION CODES:
04E6 674 : NONE
04E6 675 :
04E6 676 : SIDE EFFECTS:
04E6 677 : NONE
04E6 678 :
04E6 679 : ON ENTRY:
04E6 680 :
04E6 681 : KSP => 0
04E6 682 : 0
04E6 683 : AP
04E6 684 : FP
04E6 685 : PC
04E6 686 : 0
04E6 687 : 0
04E6 688 : AP
04E6 689 : FP
04E6 690 : SRVEXIT
04E6 691 : PC
04E6 692 : PSL
04E6 693 :
04E6 694 :-- USP => USER
CALL
FRAME

```

00000000 04E6 696 RETURN_PC:
00000000 04E6 697 .LONG 0 ; storage for user return PC
00000000 04EA 698 HANDLER_PC:
00000000 04EA 699 .LONG 0 ; storage for handler PC
04EE 700 :
04EE 701 SETUP_SUPER:
000C 04EE 702 .WORD ^M<R2,R3>
DB 04FO 703 MFPR #PR$_USP,R3 ; get the user call frame address
EE AF 10 A3 DO 04F3 704 MOVL SFSL SAVE PC(R3),B^RETURN_PC ; get the user return PC
ED AF 04 AC DO 04F8 705 MOVL 4(AP),HANDLER_PC ; save the handler address
52 OC AD DO 04FD 706 MOVL SFSL SAVE FP(FP),R2 ; get saved FP
52 00' CO 0501 707 ADDL S^#XESC CMSTKSZ,R2 ; back over change mode stack frame
62 12'AF 9E 0504 708 MOVAB B^20$,R2 ; set return address
0A F0 0508 709 INSV #<<PSL$C SUPERAPSL$S_CURMOD>+PSL$C_SUPER>,-
16 050A 710
04 A2 04 050B 711 #PSL$V_PRVMOD,-
50 01 DO 050E 712 #PSL$S_CURMOD*2 4(R2) ; set current and previous mode to super
04 0511 713 MOVL S^#SSS_NORMAL,R0 ; set correct return code
0512 714 RET ; enter super mode
20$: 715 CLRL -(SP) ; set up dummy PSL
18'AF 7E D4 0512 716 CALLG (SP),B^30$ ; create initial call frame
6E FA 0514 717 30$: .WORD ^M<> ; entry mask
0000 0518 718 PUSHL #0 ; push a dummy parameter
00 DD 051A 719 CALLS #1,W^REG_SAVE ; save the registers
0549'CF 01 FB 051C 720 MOVAL W^SM,W^MODE ; set the mode
0069'CF 012F'CF DE 0521 721 $DCLCMH S @HANDLER_PC,W^PRVHND1,#0 ; set real handler
0528 722 FAIL_CHECKNP SSS NORMAL ; check for success
0538 723 PUSHL #SSS_NORMAL
01 DD 0538 CALLS #1,W^REG_CHECKNP
05EA'CF 01 FB 053A 724 PUSHL #<<PSL$C USERAPSL$V_CURMOD>- !<PSL$C USERAPSL$V_PRVMOD>>; set return to user
03C00000 8F DD 053F 725 PUSHL RETURN_PC ; set the return PC
9E AF DD 0545 726 REI ; return to user mode
02 0548 727 .SBTTL REG_SAVE
0549 728
0549 729 :++ FUNCTIONAL DESCRIPTION:
0549 730 : Subroutine to save R2-R11 in the register save location.
0549 731 : CALLING SEQUENCE:
0549 732 PUSHL #0 ; save a dummy parameter
0549 733 CALLS #1,W^REG_SAVE ; save R2-R11
0549 734 : INPUT PARAMETERS:
0549 735 NONE
0549 736 : OUTPUT PARAMETERS:
0549 737 PUSHL #0 ; save R2-R11
0549 738 CALLS #1,W^REG_SAVE ; save R2-R11
0549 739 :--+
0549 740 :--+
0549 741 :--+
0549 742 :--+
0549 743 :--+
0549 744 :--+
0549 745 REG_SAVE:
0008'CF 14 AD 28 OFFC 0549 746 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
28 04 054B 747 MOVC3 #4*10,^X14(FP),W^REG_SAVE_AREA ; save the registers in the program
04 0552 748 RET

```

```

0553 750 .SBTTL REG_CHECK
0553 751 :++
0553 752 : FUNCTIONAL DESCRIPTION:
0553 753 : Subroutine to test R0 & R2-R11 for proper content after a service
0553 754 : execution. A snapshot is taken by the REG_SAVE routine at the
0553 755 : beginning of each step and this routine is executed after the
0553 756 : services have been executed.
0553 757 :
0553 758 : CALLING SEQUENCE:
0553 759 : PUSHL #SSS_XXXXXX ; push expected R0 contents
0553 760 : CALLS #1,W^REG_CHECK ; execute this routine
0553 761 :
0553 762 : INPUT PARAMETERS:
0553 763 : expected R0 contents on the stack
0553 764 :
0553 765 : OUTPUT PARAMETERS:
0553 766 : possible error messages printed using $PUTMSG
0553 767 :
0553 768 :--
0553 769 :
0553 770 REG_CHECK:
0553 771 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0553 772 CMPL 4(AP),R0 : is this the right fail code?
0553 773 BEQL 10$ : br if yes
0553 774 PUSHL R0 : push received data
0553 775 PUSHL 4(AP) : push expected data
0553 776 PUSHAL W^EXP : push the string variable
0553 777 CALLS #3,W^PRINT_FAIL : print the error message
0553 778 10$: CMPC3 #4*10,^X14(FP),W^REG_SAVE_AREA : check all but R0
0553 779 BEQL 20$ : br if O.K.
0553 780 SUBL3 #REG_SAVE_AREA,R3,R6 : calculate the register number
0553 781 DIVL2 #4,R6
0553 782 ADDB3 #^X2,R6,-(SP)
0553 783 BICL2 #3,R1 : set number past R0-R1 and save
0553 784 BICL2 #3,R3 : backup to register boundrys
0553 785 PUSHL (R1) : push received data
0553 786 PUSHL (R3) : push expected data
0553 787 PUSHAL W^REG : set string pntr param.
0553 788 CALLS #4,W^PRINT_FAIL : print the error message
0553 789 20$: RET
0553 790 04 0594
0553 791

```

```

0595 793 .SBTTL REG_CHECKNP
0595 794 :++
0595 795 :++ FUNCTIONAL DESCRIPTION:
0595 796 : Subroutine to test R0 & R2-R11 for proper content after a service
0595 797 : execution without printing it. A snapshot is taken by the REG_SAVE routine a
0595 798 : beginning of each step and this routine is executed after the
0595 799 : services have been executed. This routine collects the error
0595 800 : information in buffer ERLB instead of printing it.
0595 801 :
0595 802 : CALLING SEQUENCE:
0595 803 : PUSHL #SS$_XXXXXX ; push expected R0 contents
0595 804 : CALLS #1,W^REG_CHECK ; execute this routine
0595 805 :
0595 806 : INPUT PARAMETERS:
0595 807 : expected R0 contents on the stack
0595 808 :
0595 809 : OUTPUT PARAMETERS:
0595 810 : possible error messages logged in buffer ERLB which are printed
0595 811 : using routine ERLBUF_DUMP.
0595 812 :
0595 813 :--
0595 814 :
0595 815 FLAG:
00 0595 816 .BYTE 0 ; error flags are BIT0 = 0 means no errors in the bu
0596 817 ; BIT0 = 1 means errors in the buff
0596 818 ELBP: .ADDRESS ERLB ; error log buffer pointer
0000059A' 0596 819
000005EA 059A 820 ERLB: .BLKB 80 ; error log buffer
05EA 821
05EA 822 :
05EA 823 REG_CHECKNP:
0FFC 05EA 824 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
50 04 AC D1 05EC 825 CMPL 4(AP),R0 ; is this the right fail code
2B 13 05F0 826 BEQL 10$ ; br if yes
9F AF 01 88 05F2 827 BISB2 #1,FLAG ; set the error logged flag bit
52 9D AF DD 05F6 828 MOVL ELBP,R2 ; get the current error log pointer
82 03 90 05FA 829 MOVB #3,(R2)+ ; save the long word count
82 50 DD 05FD 830 MOVL R0,(R2)+ ; save received status
82 04 AC DD 0600 831 MOVL 4(AP),(R2)+ ; save expected status
82 014D'CF DE 0604 832 MOVAL W^EXP,(R2)+ ; save the string variable
62 D4 0609 833 CLRL (R2) ; set the terminator
87 AF 52 DD 060B 834 MOVL R2,ELBP ; reset the buffer pointer
004C'CF 002A'CF DE 060F 835 MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0044'CF 03 00 02 F0 0616 836 INSV #ERROR,#0,73,W^MOD_MSG_CODE ; set severity code
0008'CF 14 AD 28 29 061D 837 10$: CMPC3 #4+10,^X14(FP),W^REG_SAVE AREA ; check all but R0 and R1
3C 13 0624 838 BEQL 20$ ; br if OK
FF6A CF 01 88 0626 839 BISB2 #1,FLAG ; set error logged flag bit
52 FF67 CF DD 062B 840 MOVL ELBP,R2 ; get current error log buf pointer
82 04 90 0630 841 MOVB S^#4,(R2)+ ; set longword count
00000008'8F C3 0633 842 SUBL3 #REG_SAVE_AREA,-
56 53 0639 843 R3,R6 ; calc reg number
56 04 C6 063B 844 DIVL2 S^#4,R6 ; make it a longword count
82 56 02 C1 063E 845 ADDL3 S^#2,R6,(R2)+ ; correct for R0-R1 and save
82 61 DD 0642 846 MOVL (R1),(R2)+ ; save received results
82 63 DD 0645 847 MOVL (R3),(R2)+ ; save expected results
82 008D'CF DE 0648 848 MOVAL W^REG,(R2)+ ; save string variable

```

SATSSS43
V04-000

- SATS SYSTEM SERVICE TESTS (SUCC S.C.) 16-SEP-1984 00:54:19 VAX/VMS Macro V04-00
REG_CHECKNP J 7 5-SEP-1984 04:31:29 [UETPSY.SRC]SATSSS43.MAR;1 Page 25
(2)

FF42 CF 0044'CF 03	002A'CF 00	02	62	D4 064D 0662	850 854 20\$:	CLRL (R2) : set the terminator
0044'CF 03	002A'CF 00	02	52	DO 064F	851	MOVL R2,ELBP ; reset the buffer pointer
			DE 0654	852	MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR	; set failure message address
			FO 065B	853	INSV #ERROR,#0,73,W^MOD_MSG_CODE	; set severity code
			04 0662	855	RET	; bail out

SAT
V04

```

0663 857 .SBTTL ERLBUF_DUMP
0663 858 :++
0663 859 :++ FUNCTIONAL DESCRIPTION:
0663 860 : Routine to check for errors in the error log buffer and
0663 861 : report any that are there.
0663 862 :
0663 863 : CALLING SEQUENCE:
0663 864 : CALLS #0,W^ERLBUF_DUMP
0663 865 :
0663 866 : INPUT PARAMETERS:
0663 867 : FLAG bit 0 = 0 for no errors logged
0663 868 : FLAG bit 0 = 1 for errors logged
0663 869 : if errors logged then buffer ERLB must contain legal format errors
0663 870 :
0663 871 : OUTPUT PARAMETERS:
0663 872 : NONE
0663 873 :
0663 874 :--
0663 875 :
0663 876 ERLBUF_DUMP:
0663 877 .WORD ^M<R2,R3,R4>
0663 878 BLBC FLAG,$0$ ; br if no errors to report
0663 879 MOVAL ERLB,R2 ; set up buffer pointer
0663 880 10$: TSTL (R2) ; any more errors?
0663 881 BEQL 30$ ; br if not
0663 882 MOVZBL (R2)+,R3 ; get the longword count
0663 883 MOVL R3,R4 ; and save it
0663 884 20$: PUSHL (R2)+ ; push a parameter
0663 885 SOBGTR R3,20$ ; and push them all
0663 886 CALLS R4,W^PRINT_FAIL ; print the failure
0663 887 BRB 10$ ; do the next one
0663 888 :
0663 889 30$: MOVAL ERLB,ELBP ; reset the buffer pointer
0663 890 CLRL W^ERLB ; set fresh terminator
0663 891 RET ; bail out
0663 892 :
0663 893 :

```

SA
VO

```

0691 895 .SBTTL PRINT_FAIL
0691 896 :++
0691 897 :++ FUNCTIONAL DESCRIPTION:
0691 898 :+ Subroutine to report failures using $PUTMSG
0691 899 :+
0691 900 :+ CALLING SEQUENCE:
0691 901 :+ Mode #1 PUSHL EXPECTED Mode #2 PUSHL REG_NUMBER
0691 902 :+ PUSHL RECEIVED PUSHL EXPECTED
0691 903 :+ PUSHAL STRING VAR PUSHL RECEIVED
0691 904 :+ CALLS #3,W^PRINT_FAIL PUSHAL STRING VAR
0691 905 :+ CALLS #4,W^PRINT_FAIL PUSHL REG_NUMBER
0691 906 :+ INPUT PARAMETERS:
0691 907 :+ listed above PUSHL EXPECTED
0691 908 :+ OUTPUT PARAMETERS:
0691 909 :+ an error message is printed using $PUTMSG
0691 910 :+
0691 911 :+
0691 912 :--+
0691 913 :-
0691 914 PRINT_FAIL:
0691 915 .WORD ^M<R2,R3,R4,R5>
0693 916 $FAO_S W^CS1,W^MESSAGE,L,W^MSG,L,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
06B4 917 SPUTMSG_S W^MSGVEC ; print the message
04 6C 91 06C5 918 CMPB (AP),#4 ; is this a register message?
21 13 06C8 919 BEQL 10$ ; br if yes
06CA 920 $FAO_S W^CS2,W^MESSAGE,L,W^MSG,L,4(AP),8(AP),4(AP),12(AP)
25 11 06E9 921 BRB 20$ ; goto output message
06EB 922 10$: $FAO_S W^CS3,W^MESSAGE,L,W^MSG,L,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
0710 923 20$:
0710 924 20$:
0710 925 SPUTMSG_S W^MSGVEC ; print the message
0710 926 CALLS #0,W^MODE_ID ; identify the mode
0721 927 MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0044'CF 002A'CF 00 00 02 FB 0726 928 INSV #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
0793'CF 004C'CF 00 03 00 02 072D 929 RET
0734 930
0735 931 USERH1:
0735 932 USERH3:
0069'CF 0123'CF 0000 0735 933 .WORD 0
14 11 0737 934 MOVAL W^UM,W^MODE ; set the mode string
073E 935 BRB CEP
0740 936 SUPERH1:
0740 937 SUPERH3:
0069'CF 012F'CF 0000 0740 938 .WORD 0
09 11 0742 939 MOVAL W^SM,W^MODE ; set the mode string
0749 940 BRB CEP
074B 941 EXECH1:
074B 942 EXECH3:
0069'CF 013C'CF 0000 074B 943 .WORD 0
DE 074D 944 MOVAL W^EM,W^MODE ; set the mode string
0754 945 CEP:
0754 946 $FAO_S W^CS4,W^MESSAGE,L,W^MSG,L,MODE,#TEST_MOD_NAME ; format the error strin
0773 947 SPUTMSG_S W^MSGVEC ; print the message
0044'CF 002A'CF 00 02 FB 0784 948 MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
04 0788 949 INSV #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
0792 950 RET

```

0793 952 .SBTTL MODE_ID
0793 953 :++
0793 954 : FUNCTIONAL DESCRIPTION:
0793 955 : Subroutine to identify the mode that an exit handler is in.
0793 956 :
0793 957 : CALLING SEQUENCE:
0793 958 : CALLS #0,W^MODE_ID
0793 959 :
0793 960 : INPUT PARAMETERS:
0793 961 : MODE contains an address pointing to an ascii string desc.
0793 962 : of the current CPU mode.
0793 963 :
0793 964 : OUTPUT PARAMETERS:
0793 965 : NONE
0793 966 :
0793 967 :--
0793 968 :
0793 969 MODE_ID:
003C 0793 970 .WORD ^M<R2,R3,R4,R5>
0795 971 \$FAO_S W^CS5,W^MESSAGEL,W^MSGL,MODE ; format the error message
07AE 972 \$PUTMSG_S W^MSGVEC ; print the mode message
04 07BF 973 RET

07C0 975 MOD_MSG_PRINT:
07C0 976 :
07C0 977 : *****
07C0 978 : *
07C0 979 : * PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES
07C0 980 : * (USING THE PUTMSG MACRO).
07C0 981 : *
07C0 982 : *****
07C0 983 :
05 07C0 984 : PUTMSG <MOD_MSG_CODE,#2,TMN_ADDR,TMD_ADDR> : PRINT MSG
07DB 985 RSB ; ... AND RETURN TO CALLER
07DC 986 :
07DC 987 CHMRTN:
07DC 988 : *****
07DC 989 : *
07DC 990 : * CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER
07DC 991 : * A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED
07DC 992 : * BY THE MODE MACRO ('TO' OPTION). IT MERELY DOES
07DC 993 : * A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS
07DC 994 : * THE EFFECT OF RETURNING TO THE END OF THE MODE
07DC 995 : * MACRO EXPANSION.
07DC 996 : *
07DC 997 :*****
07DC 998 :
00000059'FF 0000 07DC 999 WORD 0 : ENTRY MASK
17 07DE 1000 JMP @CHM_CONT : RETURN TO MODE MACRO IN NEW MODE
07E4 1001 :
07E4 1002 : * RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM,' MACRO
07E4 1003 :
07E4 1004 : .END SATSSS43

SSARGS	=	00000001		PRIVMASK		00000051	R	03
SST1	=	00000004		PRVHND1		00000107	R	03
SST2	=	00000004		PRVHND2		0000010B	R	03
ARGLST	=	00000117	R	03		0000010F	R	03
BUF		000000AB	RR	03		00000050	R	03
CAN		00000085	RR	03		=	00000002	
CANEXH		0000003F	R	02		=	00000003	
CANEXHS_DESBLK	=	00000004		PSLSC_USER		=	80000000	
CANEXHS_NARGS	=	00000001		PSLSM_CM		=	00000002	
CEP		00000754	R	04		=	00000018	
CHMRTN		000007DC	RR	04		=	00000016	
CHM_CONT		00000059	RR	03		0000008D	R	03
COMP_MODE		000004B3	RR	04		0000009F	R	03
CS1		00000046	RR	02		00000553	R	04
CS2		00000078	RR	02		000005EA	R	04
CS3		000000A5	RR	02		00000549	R	04
CS4		000000D8	RR	02		00000008	R	03
CS5		0000010E	RR	02		0000005D	R	03
CURRENT_TC		00000004	RR	03		0000158	R	04
DCL		0000006D	RR	03		00004E6	R	04
DCL1		0000007D	RR	03		00000000	RG	04
DCLCMH		00000031	R	02		00000103	R	03
DCLCMHS_ADDRES	=	00000004		SERV_NAME		000004EE	R	04
DCLCMHS_NARGS	=	00000003		SETUP_SUPER				
DCLCMHS_PRVHND	=	00000008		SEVERE		=	00000004	
DCLCMHS_TYPE	=	0000000C		SFSL_SAVE_FP		=	0000000C	
DCLEXH		00000038	R	02		=	00000010	
DCLEXHS_DESBLK	=	00000004		SHRSR SHRDEF		=	00000001	
DCLEXHS_NARGS	=	00000001		SHRS_TEXT		=	00001130	
DUMMY		0000049A	R	04		SM		
ELBP		00000596	R	04		SS\$_NORMAL		
EM		0000013C	R	02		STATUS		
ERLB		0000059A	R	04		STEP		
ERLBUF_DUMP		00000663	R	04		STP0		
ERROR	=	00000002		STP1		0000003D	R	04
EXE\$C_CMSTKSZ	*****	X	04	STP10		00000078	R	04
EXEC1		000001BB	R	03		C0002D8	R	04
EXEC3		000001CF	R	03		00002ED	R	04
EXECH1		0000074B	R	04		0000329	R	04
EXECH3		0000074B	R	04		0000355	R	04
EXP		0000014D	RR	02		000000DD	R	04
FLAG		00000595	R	04		0000138	R	04
HANDLER_PC		000004EA	RR	04		0000158	R	04
HNDLR_COM		00000364	R	04		0000166	R	04
INFO	=	00000003		STP2		000001A5	R	04
LIB\$SIGNAL	*****	X	04	STP3		000001FE	R	04
MESSAGEL		000000FB	R	03		0000213	R	04
MODE		00000069	R	03		0000278	R	04
MODE_ID		00000793	R	04				
MOD_MSG_CODE		00000044	RR	03		=	0000001C	
MOD_MSG_PRINT		000007C0	RR	04		=	00000001	
MSG\$		000000A3	R	03		SUPER1		
MSGVEC		0000011F	R	03		00000193	R	03
MSGVEC1		0000012F	R	03		SUPER3		
OHC		00000113	R	03		000001A7	R	03
PR\$_USP	=	00000003		SUPERH1		00000740	R	04
PRINT_FAIL		00000691	R	04		00000740	R	04
				SUPERH3		000003E6	R	04
				SUPER_MODE		*****	GX	04
				SYSSCANEXH		*****	GX	04
				SYSSCMEXEC		*****	GX	04
				SYSSCMKRNL		*****	GX	04
				SYSDCLCMH		*****	GX	04

SY\$DCLEXH	*****	GX	04
SY\$EXIT	*****	GX	04
SY\$FAO	*****	X	04
SY\$HIBER	*****	GX	04
SY\$PUTMSG	*****	GX	04
SY\$SETPRN	*****	GX	04
SY\$WAKE	*****	GX	04
TEST	00000156	R	04
TEST_MOD_BEGIN	00000019	R	02
TEST_MOD_FAIL	0000002A	R	02
TEST_MOD_NAME	00000000	R	02
TEST_MOD_NAME_D	00000009	R	02
TEST_MOD_SUCC	0000001F	R	02
TMD_ADDR	0000004C	R	03
TMN_ADDR	00000048	R	03
TPID	00000000	R	03
UETPS_SATSMS	= 007480D9		
UETPS_TEXT	= 00741133		
UM	00000123	R	02
USER1	00000143	R	03
USER2	00000157	R	03
USER3	0000016B	R	03
USER4	0000017F	R	03
USERH1	00000735	R	04
USERH2	00000353	R	04
USERH3	00000735	R	04
USERH4	0000034C	R	04
USER_MODE	0000049D	R	04
WARNING	= 00000000		

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	0000015B (347.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
RWDATA	000001E3 (483.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SATSSS43	000007E4 (2020.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.30
Command processing	133	00:00:00.66	00:00:02.62
Pass 1	411	00:00:14.67	00:00:25.19
Symbol table sort	0	00:00:01.91	00:00:02.62
Pass 2	206	00:00:03.72	00:00:08.03
Symbol table output	18	00:00:00.14	00:00:00.34
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	809	00:00:21.19	00:00:39.13

The working set limit was 1800 pages.

88714 bytes (174 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1170 non-local and 32 local symbols.
1004 source lines were read in Pass 1, producing 28 object records in Pass 2.
52 pages of virtual memory were used to define 48 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
\$255\$DUA28:[SHRLIB]UETP.MLB;1	10
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	32
TOTALS (all libraries)	44

1367 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LI\$:\$SATSSS43/OBJ=OBJ\$:\$SATSSS43 MSRC\$:\$SATSSS43/UPDATE=(ENH\$:\$SATSSS43)+EXECML\$:/LIB+SHRLIB\$:\$UETP/LIB

0423 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

